# Windows下虚拟机搭建caffe环境

虽然在vs2013下配好了caffe、fast\_rcnn等，但还是有很多代码只能在linux下运行，不得已再配ubuntu吧。

1. 安装虚拟机

虚拟机用的是vmware11.0，文件在我移动硬盘上，网上很多破解版的地址，我就不贴了。

1. 安装ubuntu14.10

下载64位的ubuntu镜像文件：

<http://releases.ubuntu.com/14.10/ubuntu-14.10-desktop-amd64.iso>

这个是14.10版本的，之前弄了个i386，后来装cuda时提示不兼容。

15.04版的镜像：

<http://releases.ubuntu.com/15.04/ubuntu-15.04-desktop-amd64.iso>

1. 安装vmtools

安装vmtools，可以实现windows和ubuntu间的文件共享。

安装build-essentials，安装开发所需要的一些基本包

sudo apt-get install build-essential

1. 安装cuda7.0

去nvidia官网下载cuda的安装文件，地址：

<http://developer.download.nvidia.com/compute/cuda/7_0/Prod/local_installers/rpmdeb/cuda-repo-ubuntu1410-7-0-local_7.0-28_amd64.deb>

CUDA的Deb包安装较为简单，按照官网流程，事先安装必要的库

sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev

4.1 安装CUDA

然后通过以下命令获取Ubuntu 14.04 CUDA相关的repository package

sudo dpkg -i cuda-repo-ubuntu1410-7-0-local\_7.0-28\_amd64.deb

sudo apt-get update

然后开始安装CUDA Toolkit

sudo apt-get install cuda

此时需要下载较长时间，网速较慢的中途可以出去吃个饭~

4.2 环境配置

CUDA安装完毕后，需要对.bashrc加入一下命令来配置环境

export CUDA\_HOME=/usr/local/cuda-7.0

export LD\_LIBRARY\_PATH=$CUDA\_HOME/lib64:$LD\_LIBRARY\_PATH

PATH=$CUDA\_HOME/bin:$PATH

export PATH

4.3 安装CUDA SAMPLE

编译Sample文件, 整个过程大概10分钟左右

cd /usr/local/cuda-7.0/samples

sudo make

全部编译完成后， 进入 samples/bin/x86\_64/linux/release, sudo下运行deviceQuery

sudo ./deviceQuery

如果出现下列显卡信息， 则驱动及显卡安装成功：

./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 850"

CUDA Driver Version / Runtime Version 7.0 / 7.0

CUDA Capability Major/Minor version number: 3.0

Total amount of global memory: 4095 MBytes (4294246400 bytes)

( 7) Multiprocessors, (192) CUDA Cores/MP: 1344 CUDA Cores

GPU Clock rate: 1098 MHz (1.10 GHz)

Memory Clock rate: 3105 Mhz

Memory Bus Width: 256-bit

L2 Cache Size: 524288 bytes

Maximum Texture Dimension Size (x,y,z) 1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)

Maximum Layered 1D Texture Size, (num) layers 1D=(16384), 2048 layers

Maximum Layered 2D Texture Size, (num) layers 2D=(16384, 16384), 2048 layers

Total amount of constant memory: 65536 bytes

Total amount of shared memory per block: 49152 bytes

Total number of registers available per block: 65536

Warp size: 32

Maximum number of threads per multiprocessor: 2048

Maximum number of threads per block: 1024

Max dimension size of a thread block (x,y,z): (1024, 1024, 64)

Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)

Maximum memory pitch: 2147483647 bytes

Texture alignment: 512 bytes

Concurrent copy and kernel execution: Yes with 1 copy engine(s)

Run time limit on kernels: Yes

Integrated GPU sharing Host Memory: No

Support host page-locked memory mapping: Yes

Alignment requirement for Surfaces: Yes

Device has ECC support: Disabled

Device supports Unified Addressing (UVA): Yes

Device PCI Bus ID / PCI location ID: 1 / 0

Compute Mode:

< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 6.5, CUDA Runtime Version = 6.5, NumDevs = 1, Device0 = GeForce GTX 670

Result = PASS

1. 安装BLAS

Caffe的BLAS可以有三种选择，分别为atlas、mkl以及openBLAS。对于mkl可以到intel官网下载，解压完成后又一个install\_GUI.sh文件，执行该文件会出现图形安装界面，根据说明一步一步执行即可。

也可对openBLAS源码进行编译，不过需要gcc以及gfortran等相关编译器。个人认为比较便捷的是atlas，在Caffe官网上有相关的介绍，对于Ubuntu，通过以下命令可以下载atlas

sudo apt-get install libatlas-base-dev

1. 安装OpenCV 3.0.0
   1. 下载并编译[OpenCV](https://codeload.github.com/Itseez/opencv/zip/3.0.0)（官网原版OpenCV：http://opencv.org/）， 或者使用本站提供的修改版的安装包 Install-OpenCV-master （下面的安装方式使用该包完成，安装包修改了dependencies.sh文件并增加了OpenCV 3.0.0的安装文件，同时保留了原来的2.3x和2.4x版）

记得把文件考到桌面上来，不然报CMake Error: cmake\_symlink\_library: System Error: Operation not supported。

6.2. 切换到文件保存的文件夹，然后安装依赖项：

$ sudo sh Ubuntu/dependencies.sh

sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

6.3. 切换目录Ubuntu\3.0\安装OpenCV 3.0.0：

$ sudo sh opencv3\_0\_0.sh

保证网络畅通，因为软件需要联网这里时间较长，请耐心等待。。。

1. 安装 anaconda

强烈推荐使用 anaconda 的 python。它里面集成了很多包，ipython, mkl, numpy等都预装了， 省去了很多麻烦。如果有 edu 邮箱的话，还可以获得 [anaconda accelerate](https://store.continuum.io/cshop/academicanaconda)，在矩阵运算的时候，可以启用并行计算，速度快很多。

安装 anaconda:

./Anaconda-2.3.0-Linux-x86\_64.sh

安装 accelerate:

conda update conda

conda install accelerate

conda install iopro

接下来拷贝 anaconda 的许可文件到用户主目录

mv license\_academic\_20150611072013.txt ~/.continuum

然后升级 ipython, 如果不用 ipython，那就跳过下面这一步:

conda update ipython

conda update ipython-notebook

conda update ipython-qtconsole

下面测试一下 anaconda python 的功能，首先在终端下启用 ipython-notebook

$ ipython notebook

然后新建一个 ipynb 文件。在 cell 中输入

In [1]: import mkl

mkl.set\_num\_threads(4) # 设置最大线程数

mkl.get\_max\_threads() # 查看当前线程数

Out[1]: 4

可见 anaconda 已经预装了 MKL。测一下速度:

In [23]: a = np.random.random((4096, 4096))

%timeit np.dot(a,a)

1 loops, best of 3: 6.44 s per loop

1. 安装MATLAB2015a

Caffe提供了MATLAB接口， 有需要用MATLAB的同学可以额外安装MATLAB。 安装教程请自行搜索。

安装完成后添加图标 http://www.linuxidc.com/Linux/2011-01/31632.htm

sudo vi /usr/share/applications/Matlab.desktop

输入以下内容

[Desktop Entry]

Type=Application

Name=Matlab

GenericName=Matlab 2010b

Comment=Matlab:The Language of Technical Computing

Exec=sh /usr/local/MATLAB/R2010b/bin/matlab -desktop

Icon=/usr/local/MATLAB/Matlab.png

Terminal=false

Categories=Development;Matlab;

1. 安装glog

Google Logging Library（glog），下载地址：https://code.google.com/p/google-glog/，然后解压安装：

$ tar zxvf glog-0.3.3.tar.gz

$ ./configure

$ make

$ sudo make install

如果没有权限就chmod a+x glog-0.3.3 -R , 或者索性 chmod 777 glog-0.3.3 -R , 装完之后，这个文件夹就可以kill了。

1. 安装其他dependencies

对于Ubuntu 14.10，执行以下命令下载其他相关依赖库文件

$ sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libboost-all-dev libhdf5-serial-dev

$ sudo apt-get install -y libgflags-dev libgoogle-glog-dev liblmdb-dev protobuf-compiler

1. 编译Caffe，这部分参看[caffe安装手记](http://blog.csdn.net/doral/article/details/45622235)

务必记得从github上下载最新版的caffe：

<https://codeload.github.com/BVLC/caffe/zip/master>

终于完成了所有环境的配置，可以愉快的编译Caffe了！

进入caffe根目录， 首先复制一份Makefile.config

cp Makefile.config.example Makefile.config

然后修改里面的内容，主要需要修改的参数包括

CPU\_ONLY 是否只使用CPU模式，没有GPU没安装CUDA的同学可以打开这个选项

BLAS (使用intel mkl还是OpenBLAS)

MATLAB\_DIR 如果需要使用MATLAB wrapper的同学需要指定matlab的安装路径, 如我的路径为 /usr/local/MATLAB/R2013b (注意该目录下需要包含bin文件夹，bin文件夹里应该包含mex二进制程序)

DEBUG 是否使用debug模式，打开此选项则可以在eclipse或者NSight中debug程序

* 配置Makefile文件（实现对OpenCV 3.x的支持），不是Makefile.config   
  查找“Derive include and lib directories”一节，修改“LIBRARIES +=”的最后一行，增加**opencv\_imgcodecs**

opencv\_core opencv\_highgui opencv\_imgproc opencv\_imgcodecs

完成设置后， 开始编译

make all -j4

make test -j4

make runtest –j4

注意 -j4 是指使用几个线程来同时编译， 可以加快速度， j后面的数字可以根据CPU core的个数来决定， 我的CPU使4核， 所以-j4.

然后休息会，喝杯茶， 回来就差不多编译好了..

11.1. 编译Matlab wrapper

执行如下命令

make matcaffe –j4

然后就可以跑官方的matlab demo啦。

11.2. 编译Python wrapper

sudo apt-get install -y python-numpy python-scipy python-matplotlib python-sklearn python-skimage python-h5py python-protobuf python-leveldb python-networkx python-nose python-pandas python-gflags Cython ipython

make pycaffe –j4

然后基本就全部安装完拉.

接下来大家尽情地跑demo吧～

----------------------------------

12. 安装cuDNN

为了加速Caffe，可以安装cuDNN，参见这篇文章：NVIDIA CuDNN 安装说明。

# 使用MNIST数据集进行测试

Caffe默认情况会安装在$CAFFE\_ROOT，就是解压到那个目录，例如：$ home/username/caffe-master，所以下面的工作，默认已经切换到了该工作目录。下面的工作主要是，用于测试Caffe是否工作正常，不做详细评估。具体设置请参考官网：http://caffe.berkeleyvision.org/gathered/examples/mnist.html

1. 数据预处理

**$ sh data/mnist/get\_mnist.sh**

2. 重建lmdb文件。Caffe支持三种数据格式输入网络，包括Image(.jpg, .png等)，leveldb，lmdb，根据自己需要选择不同输入吧。

**$ sh examples/mnist/create\_mnist.sh**

生成mnist-train-lmdb 和 mnist-train-lmdb文件夹，这里包含了lmdb格式的数据集

3. 训练mnist

把examples/mnist/lenet\_solver.prototxt最后一行solver\_mode: GPU改成solver\_mode: CPU

**$ sh examples/mnist/train\_lenet.sh**

至此，Caffe安装的所有步骤完结，下面是一组简单的数据对比，实验来源于MNIST数据集，主要是考察一下不同系统下CPU和GPU的性能。可以看到明显的差别了，虽然MNIST数据集很简单，相信复杂得数据集，差别会更大，Ubuntu+GPU是唯一的选择了。

测试平台1：i7-4770K/16G/GTX 770/CUDA 6.5

MNIST Windows8.1 on CPU：620s

MNIST Windows8.1 on GPU：190s

MNIST Ubuntu 14.04 on CPU：270s

MNIST Ubuntu 14.04 on GPU：160s

MNIST Ubuntu 14.04 on GPU with cuDNN：30s

Cifar10\_full on GPU wihtout cuDNN：73m45s = 4428s　*（Iteration 70000）*

Cifar10\_full on GPU with cuDNN：20m7s = 1207s　*（Iteration 70000）*

测试平台2：技嘉P35X v3，i7-4720HQ@2.6G/16G/NVidia GTX 980 4G

MNIST Ubuntu 15.04 on GPU with cuDNN：33s

对比测试1：2\*E5-2620(12CPUs)/128G/Tesla K20M/CUDA5.5/CentOS 6.4

MNIST CentOS 6.4 on GPU：294s

对比测试2：Tesla K40M/CUDA6.5/ubuntu 14.04

MNIST on GPU with cuDNN：30s

对比测试3：GTX 660/CUDA6.5/ubuntu 14.04

MNIST on GPU with cuDNN：49s

对比试验1是一个不太公平的测试，毕竟性能差很大，很可能不单单是由Tesla K20s 和GTX 770带来的，也可能是因为CentOS或者是CUDA5.5(without cuDNN)的影响，但总体上的结论和Caffe官网的[reference performance numbers](http://caffe.berkeleyvision.org/performance_hardware.html" \t "_blank) 一致，对于普通用户：GTX的性价比高很多。对比试验2展现了Tesla K40的强大性能，相信对于复杂图像，它应该有更强劲的表现。（感谢香港城市大学 Ph.D Jingjing、南京理工大学 Ph.D JinLu、华中科技大学 MS LiuMaolin 提供的测试环境和测试数据。）

# 参考文献：

[**Caffe安装手记【Ubuntu15.04、CUDA-7.0、OpenBlas、OpenCV、cuDNN】**](http://blog.csdn.net/doral/article/details/45622235)

[从零安装 Caffe Install Caffe in Ubuntu 14.04 from scratch](http://coldmooon.github.io/2015/08/03/caffe_install/)

[Caffe + Ubuntu 15.04 + CUDA 7.0 新手安装配置指南](http://ouxinyu.github.io/Blogs/20140723001.html)